# Supervisor Localization of Discrete-Event Systems Based on State Tree Structures

Kai Cai and W. M. Wonham

***Abstract*—Recently we developed *supervisor localization*, a top-down approach to distributed control of discrete-event systems in the Ramadge-Wonham supervisory control framework. Its essence is the decomposition of monolithic (global) control action into local control strategies for individual agents. In this technical note, we establish a counterpart localization theory in the framework of State Tree Structures, known to be efficient for control design of very large systems. We prove that the collective localized control behavior is identical to the monolithic optimal (i.e. maximally permissive) and nonblocking controlled behavior. Further, we propose a new and more efficient localization algorithm which exploits BDD computation.**

***Index Terms*—Binary decision diagram (BDD), discrete-event systems (DES), state tree structures (STS), supervisor localization.**

## I. Introduction

Recently we developed a top-down approach, called *supervisor localization* [1]–[3], to the distributed control of discrete-event systems (DES) in the language-based Ramadge-Wonham (RW) supervisory control framework [4]. We view a plant to be controlled as comprised of independent asynchronous agents which are coupled implicitly through control specifications. To make the agents "smart" and semi-autonomous, our localization algorithm allocates *external* supervisory control action to individual agents as their *internal* control strategies, while preserving the optimality (maximal permissiveness) and nonblocking properties of the overall monolithic (global) controlled behavior. The resulting internal control strategies are typically more transparent than external supervision [2].

In this technical note and its conference precursor [16], we continue our investigation of supervisor localization, but in the (dual) *state-based* framework of DES. We adopt the recently developed formalism of *State Tree Structures* (STS) [6], [17], adapted from Statecharts [7], which has been demonstrated to be computationally efficient for monolithic (i.e., fully centralized) supervisor synthesis in the case of large systems (see [6], [17] for details). Our aim is to exploit the computational power of STS to solve distributed control problems in that case as well.

STS efficiently model hierarchical and concurrent organization of the system state set. The latter is structured as a hierarchical *state tree*, equipped with modules (*holons*) describing system dynamics. For symbolic computation, STS are encoded into predicates. A second feature contributing to computational efficiency is the use of *binary decision diagrams* (BDD) [8], a data structure which enables a compact representation of predicates that admits their logical manipulation. With BDD representation of encoded STS models, the computational complexity of supervisor synthesis becomes polynomial in the number of BDD nodes ($|\mathrm{nodes}|$), rather than in the 'flat' system state size ($|\mathrm{states}|$). In many cases $|\mathrm{nodes}| \ll |\mathrm{states}|$, thereby achieving computational efficiency. In localization, we exploit both these features of STS.

The contributions of this technical note are two-fold. First, we establish supervisor localization theory in the STS framework: formulate the state-based distributed control problem, define the concept of *control cover* for localization, and prove control equivalence between local controllers and the monolithic one. In particular, the *state-based control equivalence* between local and monolithic supervision is a new concept, which differs from the language-based one in [1]–[3] that requires the closed and marked languages respectively of local and monolithic supervision to be identical. Our second contribution is a *symbolic localization algorithm* which computes local controllers via predicates represented by BDDs. This algorithm is proved to be more efficient than that in [1]–[3]. A case study is presented in [5] which applies the symbolic localization algorithm to address distributed control of a large-scale system.

We note that [9], [10] also proposed an approach to improving transparency of supervisory control by computing *guards* (i.e. propositional formulae) for each controllable event. Starting from a set of (extended) finite automata, the approach first computes a monolithic supervisor in BDD form, then converts the BDD supervisor to guards for individual controllable events, and finally attaches the guards to the original (extended) finite automata. In converting the BDD supervisor to individual guards, several symbolic heuristic minimization techniques are used to reduce the size of guards; by contrast, our localization is based on constructing a control cover on the state set of the supervisor, and the localization algorithm works to minimize the number of cells of the control cover. Another distinction between our localization and [9], [10] is that localization determines the set of events that has to be observed for correct local decision making, while guards for controllable events determine relevant state combinations. So far our localization and the approach in [9], [10] have been applied to several different large-scale systems; it will be of interest in future work to compare the two approaches in the same settings.

We also note that [11], [12] presented a method based on extended finite state machines and "polynomial dynamic systems" to implement the monolithic supervisor by a set of distributed supervisors with communication. The approach fixes *a priori* subsets of observable events for individual agents, which may practically rule out the existence and/or global optimality of the monolithic supervisor. By contrast, our localization approach always guarantees existence and global optimality, and the observation scopes of individual agents will emerge automatically as part of the solution. In addition, the authors in [13], [14] proposed a multi-agent coordination scheme in the RW framework similar in general terms to our supervisor localization scheme. Their synthesis procedure is essentially, however, a combination of the existing standard RW supervisor synthesis with partial observation [4] and supervisor reduction [15]; and no approach is presented to handle large systems. In this technical note we establish our original supervisor localization in the STS framework, intended for large complex systems (for a case study see [5]).

The rest of the technical note is organized as follows. In Section II we provide preliminaries on STS. In Section III we formulate the distributed control problem. Section IV develops the STS supervisor localization theory and Section V presents a symbolic localization algorithm for computing local controllers. Finally in Section VI we state conclusions.

Manuscript received June 11, 2012; revised November 25, 2012 and June 28, 2013; accepted October 17, 2013. Date of publication November 06, 2013; date of current version April 18, 2014. Recommended by Associate Editor C. Hadjicostis.

The authors are with the Systems Control Group, Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4 Canada (e-mail: kai.cai@scg.utoronto.ca; wonham@control.utoronto.ca).

Color versions of one or more of the figures in this technical note are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2013.2289033

## II. Preliminaries on State Tree Structures

This section provides relevant preliminaries on the STS-based supervisory control theory, summarized from [6], [17]. A *state tree structure* (STS) $\mathbf{G}$ for modeling DES is a six-tuple

$$\mathbf{G} = (\mathbf{S}T, \mathcal{H}, \Sigma, \Delta, \mathbf{S}T_0, \mathbf{S}T_m). \tag{1}$$

Here $\mathbf{S}T$ is the *state tree* organizing the system's state set into a hierarchy; $\mathcal{H}$ is the set of *holons* (finite automata) matched to $\mathbf{S}T$ that describe the 'local' behavior of $\mathbf{G}$; $\Sigma$ is the finite event set, partitioned into the controllable subset $\Sigma_c$ and the uncontrollable subset $\Sigma_u$. Let $\mathcal{ST}(\mathbf{S}T)$ denote the set of all *sub-state-trees* of $\mathbf{S}T$. Then $\Delta : \mathcal{ST}(\mathbf{S}T) \times \Sigma \to \mathcal{ST}(\mathbf{S}T)$ is the 'global' transition function; $\mathbf{S}T_0 \in \mathcal{ST}(\mathbf{S}T)$ is the *initial* state tree; and $\mathbf{S}T_m \subseteq \mathcal{ST}(\mathbf{S}T)$ is the set of *marker* state trees. A special type of sub-state-tree of $\mathbf{S}T$ is the *basic (state) tree*, each corresponding to one 'flat' system state in the RW framework. Let $\mathcal{B}(\mathbf{S}T) \subseteq \mathcal{ST}(\mathbf{S}T)$ be the set of all basic trees of $\mathbf{S}T$. A *predicate* $P$ defined on $\mathcal{B}(\mathbf{S}T)$ is a function $P : \mathcal{B}(\mathbf{S}T) \to \{0, 1\}$ where 0 (resp. 1) stands for logical 'false' (resp. 'true'). The predicate $false(true)$ is identically 0 (1). Thus, $P$ can be identified by the subset $B_P$ of basic trees $B_P := \{b \in \mathcal{B}(\mathbf{S}T) | P(b) = 1\}$. We shall often write $b \models P$ for $P(b) = 1$. Also for a sub-state-tree $T \in \mathcal{ST}(\mathbf{S}T)$, we define $T \models P$ if and only if $(\forall b \in \mathcal{B}(T)) b \models P$. Given the *initial predicate* $P_0$ with $B_{P_0} := \{b \in \mathcal{B}(\mathbf{S}T) | b \models P_0\} = \mathcal{B}(\mathbf{S}T_0)$, and the *marker predicate* $P_m$ with $B_{P_m} := \{b \in \mathcal{B}(\mathbf{S}T) | b \models P_m\} = \bigcup_{T \in \mathbf{S}T_m} \mathcal{B}(T)$, the STS $\mathbf{G}$ in (1) can be rewritten as

$$\mathbf{G} = (\mathbf{S}T, \mathcal{H}, \Sigma, \Delta, P_0, P_m). \tag{2}$$

Next write $Pred(\mathbf{S}T)$ for the set of all predicates on $\mathcal{B}(\mathbf{S}T)$, and introduce for $Pred(\mathbf{S}T)$ the partial order $\preceq$ defined by $P \preceq P'$ iff $(\neg P) \vee P'$; namely $P \preceq P'$ holds exactly when $b \models P \Rightarrow b \models P'$ for every $b \in \mathcal{B}(\mathbf{S}T)$. Important elements in $Pred(\mathbf{S}T)$ are the reachability and coreachability predicates. Let $P \in Pred(\mathbf{S}T)$. The reachability predicate $R(\mathbf{G}, P)$ holds on just those basic trees that can be reached in $\mathbf{G}$, from some $b_0 \models P \wedge P_0$, via a path (sequence) of state trees all satisfying $P$. Dually, the coreachability predicate $CR(\mathbf{G}, P)$ is defined to hold on those basic trees that can reach some $b_m \models P \wedge P_m$ in $\mathbf{G}$ by a path of state trees all satisfying $P$. It holds that $R(\mathbf{G}, P) \preceq P$ and $CR(\mathbf{G}, P) \preceq P$. A predicate $P$ is *nonblocking* (with respect to $\mathbf{G}$) if $R(\mathbf{G}, P) \preceq CR(\mathbf{G}, P)$, i.e. every basic tree reachable from some initial state tree can also reach some marker state tree in $\mathbf{G}$.

Another key property of a predicate is controllability (cf. controllability of a language [4]). For $\sigma \in \Sigma$ define a map $M_\sigma : Pred(\mathbf{S}T) \to Pred(\mathbf{S}T)$ by $b \models M_\sigma(P)$ iff $\Delta(b, \sigma) \models P$. Thus $M_\sigma(P)$ identifies the largest subset of basic trees from which there is a one-step transition $\sigma$ into $B_P$, or at which $\sigma$ is not defined (i.e. $\Delta(b, \sigma) = \emptyset$). A predicate $P$ is called *weakly controllable* if $(\forall \sigma \in \Sigma_u) P \preceq M_\sigma(P)$. Thus $P$ is weakly controllable if it is invariant under the dynamic flow induced by uncontrollable events. For an arbitrary predicate $P \in Pred(\mathbf{S}T)$ bring in the family $\mathcal{NC}(P)$ of nonblocking and weakly controllable subpredicates of $P$, $\mathcal{NC}(P) := \{K \preceq P | \ K \text{ is nonblocking and weakly controllable}\}$. Then $\mathcal{NC}(P)$ is nonempty (since $K = false$ belongs) and is closed under arbitrary disjunctions $\vee$; in particular the supremal element $\sup \mathcal{NC}(P) := \bigvee \{K | \ K \in \mathcal{NC}(P)\}$ exists in $\mathcal{NC}(P)$.

Now define a *state feedback control* (SFBC) $f$ to be a function $f : \mathcal{B}(\mathbf{S}T) \to \Pi$, where $\Pi := \{\Sigma' \subseteq \Sigma | \Sigma_u \subseteq \Sigma'\}$. Thus $f$ assigns to each basic tree $b$ a subset of events that always contains the uncontrollable events. For $\sigma \in \Sigma$ define a *control function* $f_\sigma : \mathcal{B}(\mathbf{S}T) \to \{0, 1\}$ according to $f_\sigma(b) = 1$ iff $\sigma \in f(b)$. Thus the control action of $f$ is fully represented by the set $\{f_\sigma | \sigma \in \Sigma\}$. By definition $f_\sigma(\cdot) = true$ for every uncontrollable event $\sigma$. The closed-loop STS formed by $\mathbf{G}$ and $f$ is then written as

$$\mathbf{G}^f = \left(\mathbf{S}T, \mathcal{H}, \Sigma, \Delta^f, P_0^f, P_m^f\right) \tag{3}$$

where $P_0^f = R(\mathbf{G}^f, true) \wedge P_0$, $P_m^f = R(\mathbf{G}^f, true) \wedge P_m$, and $\Delta^f(b, \sigma) = \Delta(b, \sigma)$ if $f_\sigma(b) = 1$ and $\Delta^f(b, \sigma) = \emptyset$ otherwise. A SFBC $f$ is *nonblocking* if $R(\mathbf{G}^f, true) \preceq CR(\mathbf{G}^f, true)$.

*Theorem 1. [6, Theorem 3.2], [17]:* Let $P \in Pred(\mathbf{S}T)$ and $P_0 \wedge \sup \mathcal{NC}(P) \neq false$. Then there exists a nonblocking SFBC $f$ such that $R(\mathbf{G}^f, true) = R(\mathbf{G}, \sup \mathcal{NC}(P))$.

Theorem 1 is the main result for STS on synthesizing an optimal and nonblocking supervisor. The SFBC $f$ in Theorem 1 is represented by the control functions $f_\sigma$, $\sigma \in \Sigma$, defined by

$$f_\sigma := M_\sigma \left(\sup \mathcal{NC}(P)\right). \tag{4}$$

Thus for every $b \in \mathcal{B}(\mathbf{S}T)$, $f_\sigma(b) = 1$ if and only if $\Delta(b, \sigma) \models \sup \mathcal{NC}(P)$.

Finally, recall [4] that a finite-state automaton $\mathbf{P}$ is defined by

$$\mathbf{P} := (Q, \Sigma, \delta, q_0, Q_m) \tag{5}$$

where $Q$ is the state set, $q_0 \in Q$ is the initial state, $Q_m \subseteq Q$ is the subset of marker states, $\Sigma$ is the finite event set, and $\delta : Q \times \Sigma \to Q$ is the (partial) state transition function. In the RW (language-based) framework, a control problem is typically given in terms of a plant automaton $\mathbf{P}$ and a specification automaton $\mathbf{S}$ that imposes control requirements on $\mathbf{P}$. In setting up a control problem in STS, we can convert the pair $(\mathbf{P}, \mathbf{S})$ into an STS $\mathbf{G}$ with a predicate $P$ specifying the *illegal* basic trees that $\mathbf{G}$ is prohibited from visiting. For details of this conversion, see [6], [17]; an illustrative example is given in [5], [16].

## III. Problem Formulation

Consider a plant automaton $\mathbf{P}$ [as defined in (5)] consisting of $n$ component automata $\mathbf{P}_k, k = 1, \ldots, n$, called 'agents'. For simplicity, assume that the agents $\mathbf{P}_k$, $k = 1, \ldots, n$, are defined over pairwise disjoint alphabets,[1], i.e., $\Sigma_k \cap \Sigma_j = \emptyset$ for all $k \neq j \in [1, n]$.[2] For every $k \in [1, n]$ let $\Sigma_k = \Sigma_{c,k} \dot{\cup} \Sigma_{u,k}$, the disjoint union of the controllable event subset $\Sigma_{c,k}$ and uncontrollable event subset $\Sigma_{u,k}$. Then the plant $\mathbf{P}$ is defined over $\Sigma := \Sigma_c \dot{\cup} \Sigma_u$, where $\Sigma_c := \bigcup_{k=1}^n \Sigma_{c,k}$ and $\Sigma_u := \bigcup_{k=1}^n \Sigma_{u,k}$.

Now let a specification automaton $\mathbf{S}$ be defined over $\Sigma$, imposing a behavioral constraint on $\mathbf{P}$. As stated at the end of Section II, we convert the pair $(\mathbf{P}, \mathbf{S})$ of plant and specification into an STS $\mathbf{G} = (\mathbf{S}T, \mathcal{H}, \Sigma, \Delta, P_0, P_m)$ with a predicate $P$ specifying the illegal basic trees. The supremal nonblocking and weakly controllable subpredicate of $\neg P$ is $\sup \mathcal{NC}(\neg P)$, and we suppose that $\sup \mathcal{NC}(\neg P) \wedge P_0 \neq false$ to exclude the trivial solution. Let

$$S := R\left(\mathbf{G}, \sup \mathcal{NC}(\neg P)\right), \quad B_S := \{b \in \mathcal{B}(\mathbf{S}T) \mid b \models S\}. \tag{6}$$

Then by Theorem 1, there exists a nonblocking SFBC $f$ such that $R(\mathbf{G}^f, true) = S$, with

$$P_0^f = R(\mathbf{G}^f, true) \wedge P_0 \quad \text{and} \quad P_m^f = R(\mathbf{G}^f, true) \wedge P_m. \tag{7}$$

[1]The case where the agents may share events follows easily from the development in Section IV below, and is addressed in [5], [16]. In the language-based framework, this case is addressed in [3].

[2]Notation $[1, n] := \{1, \ldots, n\}$, the set of integers from 1 to $n$.

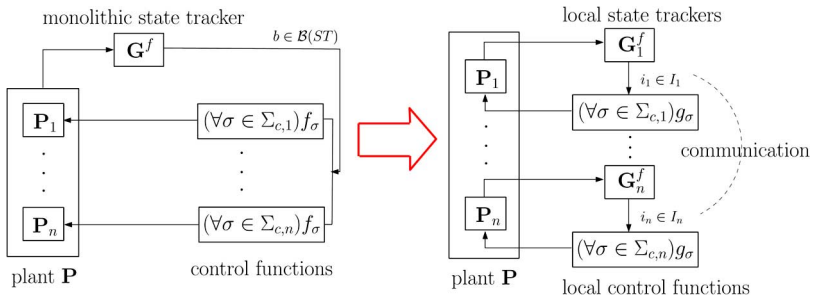


Fig. 1. Supervisor localization in STS framework.

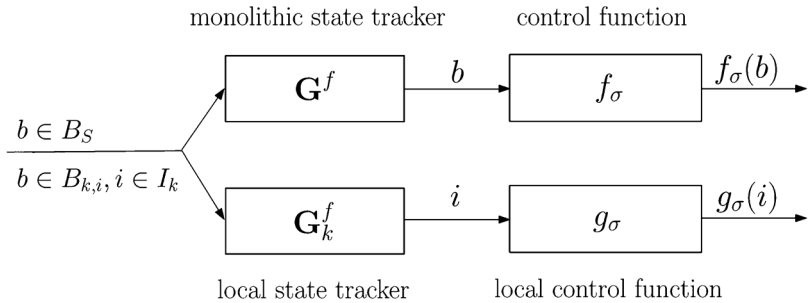


Fig. 2. Control equivalence in STS framework.

The SFBC $f$ represented by the control functions $f_\sigma, \sigma \in \Sigma$, can be written explicitly as follows: for every $b \in \mathcal{B}(\mathbf{S}T)$

$$f_\sigma(b) = \begin{cases} 1, & \text{if } \Delta(b,\sigma) = \emptyset \text{ or } \Delta(b,\sigma) \neq \emptyset \ \& \ \Delta(b,\sigma) \models S; \\ 0, & \text{if } \Delta(b,\sigma) \neq \emptyset \ \& \ \Delta(b,\sigma) \models \neg S. \end{cases} \quad (8)$$

The pair $(\mathbf{G}^f, f)$ is the monolithic optimal and nonblocking supervisor for the control problem $(\mathbf{G}, P)$, where $\mathbf{G}^f$ is the *state tracker* with state set $B_S$ which supports dynamic evolution of the controlled system, and $f$ is the SFBC which issues disablement commands based on the state where $\mathbf{G}^f$ currently resides. Since $f$ can be represented by the set of control functions $\{f_\sigma | \sigma \in \Sigma_c\}$, the supervisor $(\mathbf{G}^f, f)$ may be implemented as displayed on the left of Fig. 1 (cf. [6], [17]). Here the controllable events are grouped with respect to individual agents $\mathbf{P}_k$.

In this implementation, the state tracker $\mathbf{G}^f$ is a global entity, inasmuch as it reports each and every basic tree in $B_S$ that the system visits to all $f_\sigma$ for their decision making. For a purely distributed implementation, we propose to localize $\mathbf{G}^f$ to the individual agents so that each of them is equipped with its own local state tracker, denoted by $\mathbf{G}_k^f$, $k = 1, \ldots, n$. As will be seen in Section IV, each $\mathbf{G}_k^f$ will be constructed by finding a suitable *cover* $\mathcal{C}_k = \{B_{k,i} \subseteq B_S | i \in I_k\}$ on $B_S$; here $B_{k,i} (\neq \emptyset)$ is called a *cell* of $\mathcal{C}_k$, $I_k$ is an index set, and $\bigcup_{i \in I_k} B_{k,i} = B_S$. We will identify the index $i$ with the cell $B_{k,i}$. There will also be a set of marked cells $I_{m,k} \subseteq I_k$. Thus a local state tracker $\mathbf{G}_k^f$ reports system state evolution only in terms of cells (subsets) of basic trees, rather than singleton basic trees. This requires that the associated local control functions $g_\sigma, \sigma \in \Sigma_{c,k}$, take cells of basic trees as arguments, i.e., $g_\sigma : I_k \to \{0,1\}$. In this way each $\mathbf{G}_k^f$ tracks exactly the information sufficient for its associated $g_\sigma$ to issue correct local control. This distributed implementation is displayed on the right of Fig. 1. The use of local state trackers and local control functions may be viewed as a general implementation scheme for supervisor localization; indeed, in the language-based framework, local controllers (for individual agents in [1] and for individual controllable events in [3]) in the form of automata execute both tasks, state tracking and control decision making.

Finally, we emphasize that in the absence of monolithic tracking, the local state trackers $\mathbf{G}_k^f$ must communicate[3] in order to give correct reports on system state evolution. The communication network topology, namely who communicates with whom, is not given *a priori* but will be generated systematically as part of our localization result.

We require this distributed implementation to preserve the optimality and nonblocking properties of the monolithic supervisory control. Fix an arbitrary $k \in [1, n]$ and $\sigma \in \Sigma_{c,k}$. Suppose that the controlled system is currently visiting a basic tree $b \in B_S$; then there must exist a cell $B_{k,i}, i \in I_k$, of the cover $\mathcal{C}_k$ to which $b$ belongs. As displayed in Fig. 2, the monolithic state tracker reports $b$ to $f_\sigma$ which then makes the control decision $f_\sigma(b)$; on the other hand, a local state tracker reports (by label) the whole cell $i$ to $g_\sigma$ which then makes the control decision $g_\sigma(i)$. We say that the two pairs $(\mathbf{G}^f, f_\sigma)$ and $(\mathbf{G}_k^f, g_\sigma)$ are *control equivalent* if for every $b \in B_S$, there is $i \in I_k$ such that $b \in B_{k,i}$ (a cell of $\mathcal{C}_k$) and

$$\Delta(b,\sigma) \neq \emptyset \Rightarrow [f_\sigma(b) = 1 \text{ if and only if } g_\sigma(i) = 1]; \quad (9)$$

$$b \models P_m^f \text{ if and only if } b \models P_m \ \& \ i \in I_{m,k}. \quad (10)$$

Thus (9) requires equivalent enabling/disabling action, and (10) requires equivalent marking action. This form of control equivalence is distinct from the language-based equivalence in [1].

We now formulate the *Distributed Control Problem*. Given a plant automaton $\mathbf{P}$ [as defined in (5)] of component agents $\mathbf{P}_1, \ldots, \mathbf{P}_n$ defined over pairwise disjoint alphabets and a specification automaton $\mathbf{S}$, let $\mathbf{SUP} := (\mathbf{G}^f, \{f_\sigma | \sigma \in \Sigma_c\})$ be the corresponding STS monolithic supervisor. Construct a set of local state trackers $\mathbf{LOC}_{st} := \{\mathbf{G}_k^f | k \in [1, n]\}$, one for each agent, with a corresponding set of local control functions $\mathbf{LOC}_{cf} := \{g_\sigma, \sigma \in \Sigma_{c,k} | k \in [1, n]\}$ such that $\mathbf{LOC} := (\mathbf{LOC}_{st}, \mathbf{LOC}_{cf})$ is *control equivalent* to $\mathbf{SUP}$: that is, for every $k \in [1, n]$ and every $\sigma \in \Sigma_{c,k}$, the pairs $(\mathbf{G}^f, f_\sigma)$ and $(\mathbf{G}_k^f, g_\sigma)$ are control equivalent in the sense defined in (9) and (10).

[3]Formally, we consider that communication is by way of event synchronization; and for simplicity assume that events are communicated instantaneously, i.e. with no delay.

## IV. Supervisor Localization

We solve the Distributed Control Problem by developing a supervisor localization procedure in the STS framework. First, we need some notation from [6], [17]. Let $\sigma \in \Sigma$ and $P \in Pred(\mathbf{S}T)$. Then $\Gamma(P,\sigma)$ is the predicate which holds on the largest set of basic trees, each of which reaches a basic tree in $B_P$ by a one-step transition $\sigma$. Also $Next_{\mathbf{G}}(\sigma)$ is the predicate which holds on the largest set of basic trees of $\mathbf{G}$ that is reachable by a one-step transition $\sigma$. Define the legal subpredicate $N_{good}(\sigma)$ of $Next_{\mathbf{G}}(\sigma)$ by $N_{good}(\sigma) := Next_{\mathbf{G}}(\sigma) \wedge S$, and the illegal subpredicate $N_{bad}(\sigma) := Next_{\mathbf{G}}(\sigma) \wedge \neg S$, where $S$ is the supervisor predicate in (6).

Now fix an arbitrary $k \in [1,n]$. We develop a localization procedure which decomposes the monolithic state tracker $\mathbf{G}^f$ into a local state tracker $\mathbf{G}_k^f$ for agent $\mathbf{P}_k$ defined over $\Sigma_k$. First, we establish a *control cover* on $B_S$ (in (6)), the state set of $\mathbf{G}^f$, based solely on the control and marking information pertaining to $\Sigma_{c,k}$, as captured by the following four functions. Let $\sigma \in \Sigma_{c,k}$. Define $E_\sigma : B_S \to \{0,1\}$ by

$$E_\sigma := \Gamma\left(N_{good}(\sigma), \sigma\right) \wedge S. \tag{11}$$

Thus $E_\sigma$ is the characteristic function of the set of basic trees in $B_S$ where $\sigma$ is enabled. By this definition, for every $b \in B_S$, $b \models E_\sigma$ if and only if $\Delta(b,\sigma) \neq \emptyset$ and $f_\sigma(b) = 1$ (with $f_\sigma$ defined in (8)). Next define $D_\sigma : B_S \to \{0,1\}$ by

$$D_\sigma := \Gamma\left(N_{bad}(\sigma), \sigma\right) \wedge S. \tag{12}$$

Namely, $D_\sigma$ is the characteristic function of the set of basic trees in $B_S$ where $\sigma$ must be disabled by the supervisory control action of $S$. Thus for every $b \in B_S$, $b \models D_\sigma$ if and only if $f_\sigma(b) = 0$. Also define $M : B_S \to \{0,1\}$ according to

$$M(b) = 1 \text{ if and only if } b \models P_m^f, \quad P_m^f \text{ in (7)}. \tag{13}$$

Thus $M$ holds on the set of basic trees which are marked in $B_S$ (i.e. in $\mathbf{G}^f$). Finally define $T : B_S \to \{0,1\}$ according to

$$T(b) = 1 \text{ if and only if } b \models P_m, \quad P_m \text{ in (2)}. \tag{14}$$

So $T$ holds on the set of basic trees originally marked in $\mathbf{G}$. Note that for each $b \in B_S$, we have by $P_m^f = R(\mathbf{G}^f, true) \wedge P_m$ (in (7)) that $T(b) = 0 \Rightarrow M(b) = 0$ and $M(b) = 1 \Rightarrow T(b) = 1$. Based on the above four functions of the control and marking information of $\Sigma_{c,k}$, we define the following key binary relation $\mathcal{R}_k$ on $B_S$.

*Definition 1:* Let $\mathcal{R}_k \subseteq B_S \times B_S$. We say that $\mathcal{R}_k$ is a *control consistency relation* (with respect to $\Sigma_{c,k}$) if for every $b, b' \in B_S$, $(b,b') \in \mathcal{R}_k$ if and only if

(i) $(\forall \sigma \in \Sigma_{c,k}) E_\sigma(b) \wedge D_\sigma(b') = false = E_\sigma(b') \wedge D_\sigma(b)$;

(ii) $T(b) = T(b') \Rightarrow M(b) = M(b')$.

Informally, a pair of basic trees $(b,b')$ is in $\mathcal{R}_k$ if there is no event in $\Sigma_{c,k}$ that is enabled at $b$ but is disabled at $b'$, or vice versa (consistent disablement information); and (ii) $b$ and $b'$ are both marked or unmarked in $B_S$ provided that they are both marked or unmarked in $\mathbf{G}$ (consistent marking information). It is easily verified that $\mathcal{R}_k$ is reflexive and symmetric, but need not be transitive, and consequently not an equivalence relation (similar to the situation in [1]). This fact leads to the following definition of *control cover*. A *cover* on a set $B_S$ is a family of nonempty subsets (or *cells*) of $B_S$ whose union is $B_S$.

*Definition 2:* Let $\mathcal{C}_k = \{B_{k,i} \subseteq B_S | i \in I_k\}$ be a cover on $B_S$, with $I_k$ a suitable index set. We say that $\mathcal{C}_k$ is a *control cover* (with respect to $\Sigma_{c,k}$) if

(i) $(\forall i \in I_k, \forall b, b' \in B_{k,i})(b,b') \in \mathcal{R}_k$;

(ii) $(\forall i \in I_k, \forall \sigma \in \Sigma)[(\exists b \in B_{k,i})\Delta^f(b,\sigma) \neq \emptyset \Rightarrow (\exists j \in I_k)(\forall b' \in B_{k,i})\Delta^f(b',\sigma) \subseteq B_{k,j}]$.

A control cover $\mathcal{C}_k$ groups basic trees in $B_S$ into (possibly overlapping) cells $B_{k,i}, i \in I_k$. According to (i), all basic trees that reside in a cell $B_{k,i}$ have to be pairwise control consistent; and (ii), for each event $\sigma \in \Sigma$, all basic trees that can be reached from some basic tree in $B_{k,i}$ by a one-step transition $\sigma$ have to be covered by a certain cell $B_{k,j}$ (not necessarily unique). Hence, recursively, two basic trees $b, b'$ belong to a common cell in $\mathcal{C}_k$ if and only if (1) $b$ and $b'$ are control consistent, and (2) two future states that can be reached from $b$ and $b'$, respectively, by the same string are again control consistent. In the special case where $\mathcal{C}_k$ is a partition on $B_S$, we call $\mathcal{C}_k$ a *control congruence*.

Having defined a control cover $\mathcal{C}_k$ on $B_S$, we construct a local state tracker

$$\mathbf{G}_k^f = (I_k, \Sigma_{l,k}, \delta_k, i_{0,k}, I_{m,k}) \tag{15}$$

by the following procedure.

(P1) Each state $i \in I_k$ of $\mathbf{G}_k^f$ is a cell $B_{k,i}$ of $\mathcal{C}_k$. In particular, the initial state $i_0 \in I_k$ is a cell $B_{k,i_0}$ where the basic tree $b_0$ belongs, i.e. $b_0 \in B_{k,i_0}$, and the marker state set $I_{m,k} := \{i \in I_k | B_{k,i} \cap \{b \in B_S | b \models P_m^f\} \neq \emptyset\}$.

(P2) Choose the local event set $\Sigma_{l,k}$. For this, first define the transition function $\delta_k' : I_k \times \Sigma \to I_k$ over the entire event set $\Sigma$ by

$$\delta_k'(i,\sigma) = j \text{ if } (\exists b \in B_{k,i})\Delta^f(b,\sigma) \neq \emptyset \;\&\; (\forall b' \in B_{k,i})\Delta^f(b',\sigma) \subseteq B_{k,j}. \tag{16}$$

Then choose $\Sigma_{l,k}$ to be the union of $\Sigma_k$ of agent $\mathbf{P}_k$ with events in $\Sigma \setminus \Sigma_k$ which are *not* purely selfloop transitions of $\delta_k'$. Thus $\Sigma_{l,k} := \Sigma_k \dot{\cup} \Sigma_{com,k}$, where

$$\Sigma_{com,k} := \left\{\sigma \in \Sigma \setminus \Sigma_k \mid (\exists i, j \in I_k)\; i \neq j \;\&\; \delta_k'(i,\sigma) = j\right\}. \tag{17}$$

The set $\Sigma_{com,k}$ is the subset of communication events of other agents $\mathbf{P}_j$ $(j \neq k)$ that $\mathbf{P}_k$ needs to 'know' for its local decision making.[4]

(P3) Define the transition function $\delta_k$ to be the restriction of $\delta$ to $\Sigma_{l,k}$, namely $\delta_k := \delta|_{\Sigma_{l,k}} : I_k \times \Sigma_{l,k} \to I_k$.

Thus the above constructed local state tracker $\mathbf{G}_k^f$ is an automaton, which reports system state evolution in terms of cells (subsets) of basic trees which are crucial for, and only for, the local control and marking with respect to $\Sigma_{c,k}$ of agent $\mathbf{P}_k$. Owing to the possible overlapping of cells in $\mathcal{C}_k$, the choices of $i_0$ and $\delta_k$ may not be unique, and consequently $\mathbf{G}_k^f$ may not be unique. In that case we select an arbitrary instance of $\mathbf{G}_k^f$. Clearly if $\mathcal{C}_k$ happens to be a control congruence, then $\mathbf{G}_k^f$ is unique.

Finally, we define *local control functions* $g_\sigma, \sigma \in \Sigma_{c,k}$, to be compatible with $\mathbf{G}_k^f$. Let $\sigma \in \Sigma_{c,k}$. Define $g_\sigma : I_k \to \{0,1\}$ by

$$g_\sigma(i) = 1 \text{ if and only if } (\exists b \in B_{k,i})\; b \models \Gamma\left(N_{good}(\sigma), \sigma\right). \tag{18}$$

So $g_\sigma$ enables $\sigma$ at state $i$ of $\mathbf{G}_k^f$ whenever there is a basic tree in the cell $B_{k,i}$ at which $\sigma$ is enabled.

We have now completed the localization procedure for an arbitrarily chosen agent $\mathbf{P}_k$, $k \in [1,n]$. The procedure is summarized and illustrated in Fig. 3. Applying the same procedure for every agent, we obtain a set of local state trackers $\mathbf{LOC}_{st} := \{\mathbf{G}_k^f | k \in [1,n]\}$ with a corresponding set of local control functions $\mathbf{LOC}_{cf} := \{g_\sigma, \sigma \in \Sigma_{c,k} | k \in [1,n]\}$. Our main result, below, states that this pair $\mathbf{LOC} := (\mathbf{LOC}_{st}, \mathbf{LOC}_{cf})$ is a solution to the Distributed Control Problem.

*Theorem 2:* The pair $\mathbf{LOC} := (\mathbf{LOC}_{st}, \mathbf{LOC}_{cf})$ of local state trackers and local control functions is control equivalent to the optimal

[4] In the language-based framework, a set of communication events similar to $\Sigma_{com,k}$ is given in [3].

and nonblocking supervisor $\mathbf{SUP} := (\mathbf{G}^f, \{f_\sigma | \sigma \in \Sigma_c\})$; namely, for every $k \in [1,n]$, $\sigma \in \Sigma_{c,k}$, and $b \in B_S$, there exists $i \in I_k$ such that $b \in B_{k,i}$ and

(i) $\Delta(b,\sigma) \neq \emptyset \Rightarrow [f_\sigma(b) = 1$ if and only if $g_\sigma(i) = 1]$;

(ii) $b \models P_m^f$ if and only if $b \models P_m \ \& \ i \in I_{m,k}$.

*Proof:* Let $k \in [1,n]$, $\sigma \in \Sigma_{c,k}$, and $b \in B_S$. Then there must exist a state $i \in I_k$ of the tracker $\mathbf{G}_k^f$, corresponding to a cell $B_{k,i}$ of the control cover $\mathcal{C}_k$, such that $b \in B_{k,i}$. For (i), suppose that $\Delta(b,\sigma) \neq \emptyset$; it will be shown that $f_\sigma(b) = 1$ if and only if $g_\sigma(i) = 1$. (If) Let $g_\sigma(i) = 1$, i.e. there is $b' \in B_{k,i}$ such that $b' \models \Gamma(N_{good}(\sigma), \sigma)$. Since $b'$ is also in $B_S$, we have $b' \models \Gamma(N_{good}(\sigma),\sigma) \wedge S = E_\sigma$. It follows from $b \in B_{k,i}$ that $(b,b') \in \mathcal{R}_k$ and $E_\sigma(b') \wedge D_\sigma(b) \equiv false$. Hence $D_\sigma(b) \equiv false$, which is equivalent to $f_\sigma(b) = 1$ by the definition of $D_\sigma$ in (12). (Only if) Let $f_\sigma(b) = 1$. Since $\Delta(b,\sigma) \neq \emptyset$ and $b$ is in $B_S$, we have by the definition of $E_\sigma$ in (11) that $b \models E_\sigma = \Gamma(N_{good}(\sigma),\sigma) \wedge S$. We then conclude from $b \in B_{k,i}$ and the definition of $g_\sigma$ in (18) that $g_\sigma(i) = 1$.

Now we show (ii). (If) Let $b \models P_m$ (i.e. $T(b) = 1$) and $i \in I_{m,k}$. Then there is $b' \in B_{k,i}$ such that $b' \models P_m^f$; so $M(b') = 1$, and also $T(b') = 1$. Since $(b,b') \in \mathcal{R}_k$ and $T(b) = T(b')$, we have $M(b) = M(b') = 1$. (Only if) Let $b \models P_m^f$ (i.e. $M(b) = 1$). Then $T(b) = 1$, i.e. $b \models P_m$, and also $i \in I_{m,k}$ by the construction of the tracker $\mathbf{G}_k^f$. □

In essence Theorem 2 asserts that every set of control covers generates a solution to the Distributed Control Problem. In fact, the converse statement is true: every solution to the Distributed Control Problem is generated by a suitable set of control covers. The latter is formulated and proved in [5].

## V. Symbolic Localization Algorithm

In this section we design an STS localization algorithm for computing local controllers, which is more efficient than the counterpart algorithm in [1].

It is desirable to have an efficient algorithm which computes a set of control covers that yields a set of *state-minimal* local state trackers (possibly non-unique); the latter is, by Theorem 2, a solution to the Distributed Control Problem. The minimal state problem is, however, known to be NP-hard [15]. Nevertheless, a polynomial-time localization algorithm was proposed in [1] which generates a control congruence (instead of a control cover), and empirical evidence [2] shows that significant state size reduction can often be achieved. In the following, we propose a new localization algorithm which is based on STS. The advantage of using STS is that the efficiency of the new algorithm is improved compared to the one in [1], as will be shown below.

We sketch the idea of the algorithm as follows. Let $B_S$ in (6) be labeled as $B_S = \{b_0, \ldots, b_{N-1}\}$, and $\Sigma_{c,k} \subseteq \Sigma_c$ be the controllable events of agent $\mathbf{P}_k$, $k \in [1,n]$. Our algorithm will generate a control congruence $\mathcal{C}_k$ on $B_S$ (with respect to $\Sigma_{c,k}$). This is done symbolically. First introduce the set $\tilde{B}_S = \{\tilde{b}_0, \ldots, \tilde{b}_{N-1}\}$, where $\tilde{b}_i : \mathcal{B}(\mathbf{ST}) \to \{0,1\}$ are predicates defined by $\tilde{b}_i(b) = 1$ if and only if $b = b_i$. Two elements of $\tilde{B}_S$ may be merged (by "$\vee$") if (i) their corresponding basic trees are control consistent (line 10 in the pseudocode below, where $\tilde{\mathcal{R}}_k : Pwr(B_S) \to \{0,1\}$ is defined by $B_1 \models \tilde{\mathcal{R}}_k$ if and only if $(\forall b, b' \in B_1)(b,b') \in \mathcal{R}_k$); and (ii) all corresponding downstream basic trees reachable from $b, b'$ by identical strings are also control consistent (line 12, where $\tilde{\Delta} : Pred(\mathbf{ST}) \times \Sigma \to Pred(\mathbf{ST})$ is the predicate counterpart of $\Delta$ in (1), the global transition function of STS). This $\tilde{\Delta}$ operator is the key to the improved efficiency of our STS-based algorithm: since $\tilde{\Delta}$ can handle one-step transitions of a predicate corresponding to a subset of basic trees, in each call of the CHECK_MERGE function we may also check control consistency by applying $\tilde{\mathcal{R}}_k$ to this subset; this is more efficient than the algorithm in [1] which in each call of the CHECK_MERGE function checks control consistency only for a pair of flat states (corresponding to basic trees). Finally, after checking all the elements in $\tilde{B}_S$, the algorithm at line 8 generates a control congruence $\mathcal{C}_k$ each cell of which consists of the basic trees $b_i$ whose corresponding predicates $\tilde{b}_i$ are merged together in $\tilde{B}_S$.

*Theorem 3:* The STS localization algorithm terminates, has (worst-case) time complexity $O(N^3)$, and the generated $\mathcal{C}_k$ is a control congruence on $B_S$.

We remark that the STS localization algorithm realizes the same functionality as the one in [1], and moreover improves the time complexity from $O(N^4)$ in [1] to $O(N^3)$. This is achieved by the fact that the (global) transition function of STS can handle subsets of basic trees simultaneously, which makes checking the control consistency relation in each call of the CHECK_MERGE function more efficient.

The following is the pseudocode of the algorithm. Notation: "\" denotes set subtraction; $x \prec y$ means $x \preceq y$ and $x \neq y$.

1: **procedure** MAIN()

2: **for** $i := 0$ to $N-2$ **do**

3: **for** $j := i+1$ to $N-1$ **do**

4: $B = \tilde{b}_i \vee \tilde{b}_j$;

5: $W = \emptyset$;

6: **if** Check_Merge$(B, W, i, \tilde{B}_S) = true$ **then**

7: $\tilde{B}_S = (\tilde{B}_S \cup W) \setminus \{\tilde{b} \in \tilde{B}_S | (\exists w \in W)\tilde{b} \prec w\}$;

8: **return** $\mathcal{C}_k = \{\cup_i b_i | \vee_i \tilde{b}_i \in \tilde{B}_S\}$;

9: **function** CHECK_MERGE$(B, W, i, \tilde{B}_S)$

10: **if** $\{b \in \mathcal{B}(\mathbf{ST}) | b \models B\} \not\models \tilde{\mathcal{R}}_k$ **then return** $false$;

11: $W = (W \cup B) \setminus \{w \in W | w \prec B\}$;

12: **for** each $\sigma \in \Sigma$ with $\tilde{\Delta}(B,\sigma) \wedge S \neq false$ **do**

13: **if** $(\tilde{\Delta}(B,\sigma) \wedge S) \preceq w$ for some $w \in W \cup \tilde{B}_S$ **then continue**;

14: **if** $(\tilde{\Delta}(B,\sigma) \wedge S) \wedge \tilde{b}_r \neq false$ for some $r < i$ **then return** $false$;

15: $B = (\tilde{\Delta}(B,\sigma) \wedge S) \vee (\bigvee \{w | w \in W \& w \wedge (\tilde{\Delta}(B,\sigma) \wedge S) \neq false\})$;

16: **if** Check_Merge$(B, W, i, \tilde{B}_S) = false$ **then return** $false$;

17: **return** $true$;

*Proof of Theorem 3:* Since both $B$ at line 4 and $\tilde{\Delta}(B,\sigma) \wedge S$ at line 15 are the join "$\vee$" of the predicates in $\tilde{B}_S$, so is each element of $W$ which is updated only at line 11. Thus, the size of $\tilde{B}_S$, which is updated only at line 7, is non-increasing. Because the initial size $N$ is finite, the algorithm must terminate. In the worst case, there can be $N(N-1)/2$ calls (by lines 2, 3) made to the function CHECK_MERGE, which can then make $N$ calls (by lines 12, 13) to itself. So the worst-case time complexity is $N^2(N-1)/2 = O(N^3)$.

It is left to show that $\mathcal{C}_k$ generated at line 8 is a control congruence. First, the control consistency of every pair of basic trees in the same cell of $\mathcal{C}_k$ is guaranteed by the check at line 10; so $\mathcal{C}_k$ is a control cover. Second, the set subtraction "\" when updating $W$ at line 11 and $\tilde{B}_S$ at line 7 ensures that the cells of $\mathcal{C}_k$ are pairwise disjoint; thus $\mathcal{C}_k$ is a partition on $B_S$. Therefore, $\mathcal{C}_k$ is a control congruence. □

*Example 1:* We provide an example, displayed in Fig. 4, to illustrate the STS localization algorithm. Initially, $\tilde{B}_S = \{\tilde{b}_0, \tilde{b}_1, \tilde{b}_2, \tilde{b}_3\}$. The ranges of indices $i$ and $j$ at lines 2 and 3 are $i \in [0,2]$ and $j \in [i+1, 3]$.

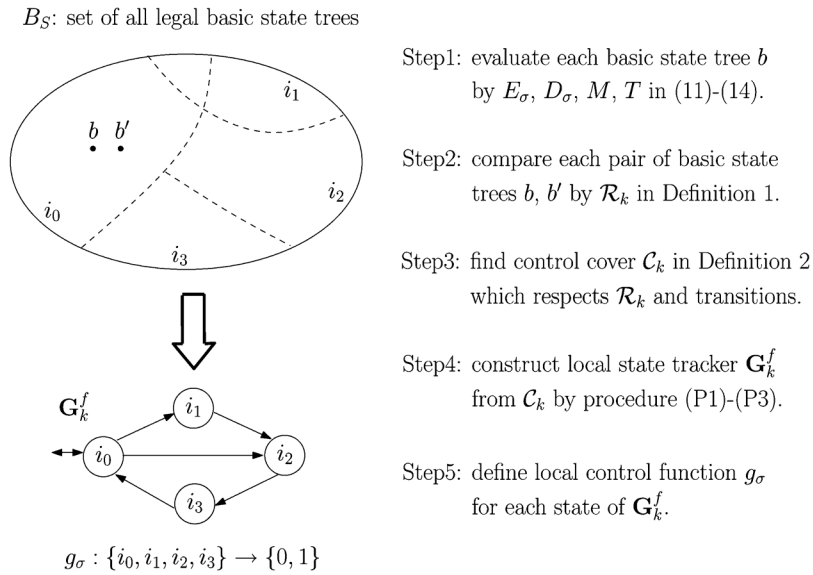


Fig. 3. Supervisor localization procedure.

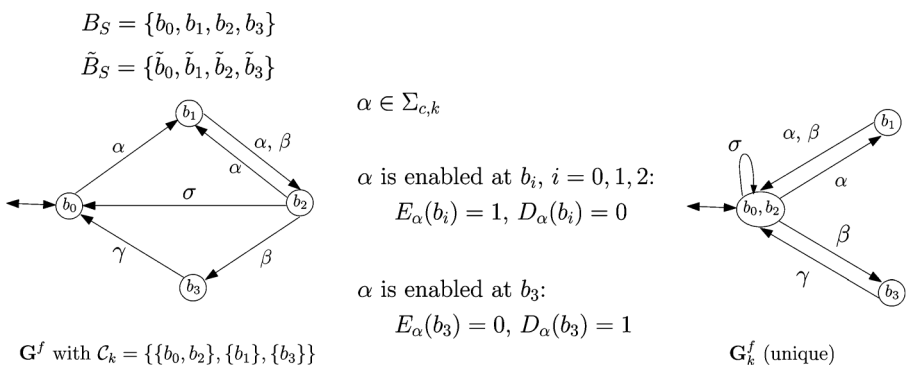


Fig. 4. Example: STS localization algorithm.

1) $(\tilde{b}_0, \tilde{b}_1)$ cannot be merged. First, $B = \tilde{b}_0 \vee \tilde{b}_1$ and the test at line 10 is passed since $\{b_0, b_1\} \models \tilde{\mathcal{R}}_k$; so $W = \tilde{b}_0 \vee \tilde{b}_1$. Second, $B$ is updated at line 15 to $B = \tilde{b}_0 \vee \tilde{b}_1 \vee \tilde{b}_2$ and the test at line 10 is still passed since $\{b_0, b_1, b_2\} \models \tilde{\mathcal{R}}_k$; so $W = \tilde{b}_0 \vee \tilde{b}_1 \vee \tilde{b}_2$. Third, $B$ is updated at line 15 to $B = \tilde{b}_0 \vee \tilde{b}_1 \vee \tilde{b}_2 \vee \tilde{b}_3$ but now the test at line 10 fails since $\{b_0, b_1, b_2, b_3\} \not\models \tilde{\mathcal{R}}_k$ (indeed, $(b_i, b_3) \notin \mathcal{R}_k$, $i = 0, 1, 2$). Note that when $B = \tilde{b}_0 \vee \tilde{b}_1 \vee \tilde{b}_2$ the global transition function $\tilde{\Delta}(B, \sigma)$ at lines 12–15 handles the local transitions at basic trees $b_1, b_2, b_3$ simultaneously:

$$\tilde{\Delta}(\tilde{b}_0 \vee \tilde{b}_1 \vee \tilde{b}_2, \alpha) = \tilde{b}_0 \vee \tilde{b}_1 \vee \tilde{b}_2 \;\&\; \tilde{\Delta}(\tilde{b}_0 \vee \tilde{b}_1 \vee \tilde{b}_2, \beta) = \tilde{b}_2 \vee \tilde{b}_3.$$

This operation is more efficient than the localization algorithm in [1]; there, only a pair of basic trees of $\{b_1, b_2, b_3\}$ and the associated transitions can be processed at a single step.

2) $(\tilde{b}_0, \tilde{b}_2)$ can be merged. First, $B = \tilde{b}_0 \vee \tilde{b}_2$ and the test at line 10 is passed since $\{b_0, b_2\} \models \tilde{\mathcal{R}}_k$; so $W = \tilde{b}_0 \vee \tilde{b}_2$. Second, $B$ is updated at line 15 to $B = \tilde{b}_1$ and the test at line 10 is trivially passed; so $W = \{\tilde{b}_0 \vee \tilde{b}_2, \tilde{b}_1\}$. Now one verifies that the condition at line 13 is satisfied for both transitions $\alpha$ and $\beta$ defined at $b_1$, so the "for"-loop from line 12 to line 16 is finished without calling the CHECK_MERGE function. Hence $true$ is returned at line 6 and $\tilde{B}_S$ is updated at line 7 to $\tilde{B}_S = \{\tilde{b}_0 \vee \tilde{b}_2, \tilde{b}_1, \tilde{b}_3\}$.

3) $(\tilde{b}_0, \tilde{b}_3)$ cannot be merged because $(b_0, b_3) \notin \mathcal{R}_k$ and the test at line 10 fails.

4) $(\tilde{b}_1, \tilde{b}_2)$ cannot be merged. First, $B = \tilde{b}_1 \vee \tilde{b}_2$ and the test at line 10 is passed since $\{b_1, b_2\} \models \tilde{\mathcal{R}}_k$; so $W = \tilde{b}_1 \vee \tilde{b}_2$. Second, $B$ is updated at line 15 to $B = \tilde{b}_1 \vee \tilde{b}_2 \vee \tilde{b}_3$ but the test at line 10 fails since $\{b_1, b_2, b_3\} \not\models \tilde{\mathcal{R}}_k$.

5) $(\tilde{b}_1, \tilde{b}_3)$ cannot be merged because $(b_1, b_3) \notin \mathcal{R}_k$ and the test at line 10 fails.

6) $(\tilde{b}_2, \tilde{b}_3)$ cannot be merged because $(b_2, b_3) \notin \mathcal{R}_k$ and the test at line 10 fails.

Finally, $\tilde{B}_S = \{\tilde{b}_0 \vee \tilde{b}_2, \tilde{b}_1, \tilde{b}_3\}$ and line 8 generates a control congruence $\mathcal{C}_k = \{\{b_0, b_2\}, \{b_1\}, \{b_3\}\}$. The local state tracker (unique in this case) constructed from $\mathcal{C}_k$ is displayed in Fig. 4.

## VI. Conclusion

We have developed state-based supervisor localization in the STS framework. In this localization scheme, each agent is endowed with its own local state trackers and local control functions, while being coordinated with its fellows through event communication in such a way that the collective local control action is identical to the global optimal and nonblocking action. Compared to the language-based RW counterpart [1], we have designed a more efficient symbolic localization algorithm by exploiting BDD computation.

## REFERENCES


[1] K. Cai and W. M. Wonham, "Supervisor localization: A top-down approach to distributed control of discrete-event systems," *IEEE Trans. Autom. Control*, vol. 55, no. 3, pp. 605–618, Mar. 2010.

[2] K. Cai and W. M. Wonham, "Supervisor localization for large discrete-event systems—Case study production cell," *Int. J. Adv. Manufact. Technol.*, vol. 50, no. 9–12, pp. 1189–1202, 2010.

[3] K. Cai and W. M. Wonham, "New results on supervisor localization, with application to multi-agent formations," in *Proc. Workshop Discrete-Event Syst.*, Guadalajara, Mexico, 2012, pp. 233–238.

[4] W. M. Wonham, "Supervisory control of discrete-event systems," Systems Control Group, ECE Dept., Univ. Toronto, Toronto, ON, Canada, Tech. Rep., Jul. 1, 2013 [Online]. Available: http://www.control.toronto.edu/DES.

[5] K. Cai and W. M. Wonham, "Supervisor localization of discrete-event systems based on state tree structures," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2013 [Online]. Available: http://arxiv.org/abs/1306.5441

[6] C. Ma and W. M. Wonham, *Nonblocking Supervisory Control of State Tree Structures*. New York: Springer-Verlag, 2005.

[7] D. Harel, "Statecharts: A visual formalism for complex systems," *Sci. Comp. Programming*, vol. 8, no. 3, pp. 231–274, 1987.

[8] R. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Computers*, vol. C-35, no. 8, pp. 677–691, 1986.

[9] S. Miremadi, K. Akesson, and B. Lennartson, "Symbolic computation of reduced guards in supervisory control," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 4, pp. 754–765, 2011.

[10] S. Miremadi, B. Lennartson, and K. Akesson, "A BDD-based approach for modeling plant and supervisor by extended finite automata," *IEEE Trans. Control Syst. Technol.*, vol. 20, no. 6, pp. 1421–1435, 2012.

[11] Y. Yang, A. Mannani, and P. Gohari, "Implementation of supervisory control using extended finite-state machines," *Int. J. Syst. Sci.*, vol. 39, no. 12, pp. 1115–1125, 2008.

[12] A. Mannani and P. Gohari, "Formal modeling and synthesis of statetransferring (event-transferring) communication among decentralized supervisors for discrete-event systems," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, San Antonio, TX, 2009, pp. 3231–3242.

[13] K. T. Seow, M. T. Pham, C. Ma, and M. Yokoo, "Coordination planning: Applying control synthesis methods for a class of distributed agents," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 2, pp. 405–415, 2009.

[14] M. T. Pham and K. T. Seow, "Discrete-event coordination design for distributed agents," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 1, pp. 70–82, 2012.

[15] R. Su and W. M. Wonham, "Supervisor reduction for discrete-event systems," *Discrete Event Dyn. Syst.*, vol. 14, no. 1, pp. 31–53, 2004.

[16] K. Cai and W. M. Wonham, "Supervisor localization of discrete-event systems based on state tree structures," in *Proc. 51st IEEE Conf. Decision Control*, Maui, HI, 2012, pp. 5822–5827.

[17] C. Ma and W. M. Wonham, "Nonblocking supervisory control of state tree structures," *IEEE Trans. Autom. Control*, vol. 51, no. 5, pp. 782–793, 2006.